Санкт-Петербургский государственный университет
Кафедра компьютерных технологий и систем

Холод Николай Григорьевич

Выпускная квалификационная работа бакалавра

Сопоставление видеопотока с цифровыми изображениями земной поверхности

Направление 010400
Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Коровкин М. В.

Санкт-Петербург
2017

Содержание

Введение

В настоящее время спутниковые карты постоянно совершенствуются и представляют всё больше и больше информации пользователям. В связи с этим остро встает проблема, которая состоит в том, что карты необходимо постоянно обновлять для того, чтобы они содержали как можно более актуальную информацию о местности. Также очень часто возникает задача привязки движения некоторого летательного объекта к карте, например в ситуациях, когда данные GPS могут быть недоступны. В связи с этими двумя вышеописанными проблемами возникает задача сопоставления видеопотока с цифровыми изображениями земной поверхности.

Данная задача интересна тем, что условия, в которых производится видеосъемка местности, существенно отличаются от условий, в которых получено изображение карты, поэтому стандартные методы на основе ключевых точек к ней неприменимы. Кроме того, изображение карты может быть схемой, что еще больше усложняет задачу, поскольку невозможно напрямую пользоваться информацией об яркости изображения.

Данную задачу можно разбить на две подзадачи. Первая подзадача — это склейка кадров видеопотока в одну общую панораму. Два возможных метода решения этой задачи рассмотрены в главе 1. Вторая подзадача — наложение двух различных изображений друг на друга. В данной работе она решается путем нахождения преобразования между скелетными графами дорог обоих изображений. В главе 2 рассмотрен алгоритм построения скелетного графа, а в главе 3 описан метод нахождения преобразования подобия между двумя графами.

Постановка задачи

Пусть имеется видеопоток — последовательность кадров, снятых одной камерой, на которых изображена некоторая местность. Внутренние параметры камеры неизвестны, однако требуется гарантия отсутствия эффектов рыбьего глаза на всех изображениях. Кроме того, камера должна быть направлена строго вниз.

Также имеется другое изображение этой же местности, будем называть его картой. К местности, на которой происходит съемка предъявляется одно требование, она должна содержать как минимум два перекрестка, которые хорошо просматриваются как на карте, так и на видео.

Задача состоит в том, чтобы для каждого кадра видеопотока определить регион на карте, который изображен на этом кадре. Если кадры очень близки друг к другу, допускается выбрать опорные кадры и производить привязку только для них.

Целью моей работы является построение системы, решающей эту задачу с как можно большей точностью. Данная задача была разбита на подзадачи следующим образом:

1. Построение панорамы.
2. Нахождение дорог и их скелетных графов.
3. Нахождение преобразования между двумя скелетными графами.

Обзор литературы

Тема данной работы тесно связана с задачей сопоставления карты и изображения.

В [3] описан способ решения задачи для спутниковых снимков. Производится сегментация входных изображений с использованием адаптивного порога, который вычисляется на основе заданного пользователем параметра T, среднего значения и среднеквадратичного отклонения яркости в заданном окне. В качестве признаков используются моменты, инвариантные к аффинному преобразованию — cross weighted moments. Сопоставление производится при помощи поиска в глубину с возвратом в дереве, содержащем признаки объектов, полученных при сегментации с разными значениями параметра T. Внутри алгоритма поиска также учитываются дополнительные унарные и бинарные ограничения. На этом этапе происходит нахождение глобального преобразования между объектами. Заключительным этапом является нахождение локальных преобразований между соответствующими объектами на изображениях.

В [1] описан способ решения с использованием модифицированной частичной метрики Хаусдорфа. На обоих изображениях выделяются контуры, после чего происходит преобразование двумерных изображений точек границ в одномерную последовательность точек, путем обхода изображения по кривой Гильберта. Полученная последовательность разбивается на равномерные интервалы, каждому интервалу соответствует особая точка. Искомое преобразование вычисляется в виде оптимального аффинного преобразования, переводящего один набор точек в другой. В работе представлены результаты тестирования алгоритма на сдвинутых друг относительно друга изображениях.

Немного модифицированный подход использовался в работе [2]. Связь между изображениями искалась в виде Евклидового преобразования. Трехмерное пространство параметров $(\Delta x, \Delta y, \theta)$ сводилось к одномерному при помощи Гильбертового обхода. Далее оптимальные параметры вычислялись на основе метрики близости, которая использует информацию о модуле и направлении градиента точек изображения. Алгоритм хорошо работает при малых углах поворота $\theta < 15^\circ$.

Глава 1. Склейка панорамы

Первый этап алгоритма — склейка изображений видеопотока в одно панорамное изображение. Она производится по нескольким причинам: во-первых, на панорамном изображении можно найти больше признаков, чем на отдельных изображениях видеопотока. Во-вторых, появляется возможность определить положение кадра, даже если он не содержит существенных особенностей, так как известно положение этого кадра относительно других.

Поскольку для нахождения преобразования между изображениями используется информация о дорогах, то вместо того, чтобы строить полноценную цветную панораму и искать на ней дороги, будем производить бинаризацию каждого кадра в отдельности и склеивать результаты в общую карту дорог. Такой подход позволяет повысить качество найденных дорог и вместе с тем повысить производительность, так как отпадает необходимость в интерполяции яркостей пикселей, которая обычно применяется для того, чтобы результат оставался гладким. Бинаризация подробно описана в главе 2.

Алгоритм построения панорамы заключается в следующем: рассматриваются пары соседних кадров, считая положение одного кадра в панораме известным, находится положение другого кадра, с помощью алгоритма, основанного на методе, описанном в [6]. Расположение кадра в панораме задается сеткой: изображение разделяется на квадратные участки известного размера, каждый квадрат изображения переходит в некоторый четырехугольник панорамного изображения при помощи билинейного преобразования. Положение первого кадра в панораме выбирается произвольно. Координаты вершин четырехугольников, которые являются координатами узлов сетки, определяют положение кадра в панораме. Соседними считаются кадры, время которых отличается на Δt, то есть рассматриваются кадры $t_0, t_0 + \Delta t, t_0 + 2\Delta t, \ldots$ и так далее. Для каждого кадра выполняется следующая последовательность действий:

1. Предварительная обработка: ресемплинг, перевод изображения в оттенки серого.

2. Слежение за ключевыми точками, которые были найдены на предыду-

щем кадре.

3. Нахождение примерного преобразования гомографии между двумя соседними кадрами.
4. Улучшение качества панорамы при помощи метода content-preserving warping.
5. Нахождение ключевых точек, эти точки будут прослеживаться на следующем кадре.

1.1 Нахождение особых точек и их прослеживание

Для нахождения ключевых точек использовалась реализация метода, описанного в [7], из библиотеки OpenCV [8]. Алгоритм находит наиболее четкие углы на изображениях. В библиотеке OpenCV реализация алгоритма представлена в функции goodFeaturesToTrack, которая выполняет следующее:

1. Вычисляет меру качества угла для всех пикселов изображения.
2. Находит локальные максимумы меры с окне 3x3.
3. Отбрасывает углы, у которых мера качества меньше заданного порога.
4. Оставшиеся углы сортируется по убыванию меры качества.
5. Если в окрестности угла есть угол с большим качеством, то угол с меньшим качеством отбрасывается.

Для слежения за найденными особыми точками используется алгоритм оптического потока Лукаса-Канаде [9]. Он основан на разложении в ряд Тейлора функции яркости изображения $I(x + \Delta x, y + \Delta y, t + \Delta t)$ для пикселов в окне с центром в особой точке в окрестности точки (x, y, t). Приравняв интенсивности пикселов на изображениях $I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x, y, t)$ для всех пикселов в окне, получим систему линейных уравнений, которую можно решить методом наименьших квадратов и таким образом найти координаты особой точки на втором снимке. Для более точного определения оптического потока используется пирамида изображений разного масштаба, каждый следующий уровень отличается от предыдущего уменьшением масштаба изображения в два раза. Поток вычисляется начиная от меньшего изображения к большему, используя поток на нижних уровнях

как начальное приближение для потока верхних уровнях. Использование оптического потока является более быстрым способом нахождения соответствий, чем использование дескрипторов особых точек, однако такой подход хорошо работает только при небольших смещениях между кадрами и незначительных изменениях условий освещенности, но поскольку в данной задаче происходит работа с видеопотоком, то этот алгоритм подходит для использования в рамках этой задачи.

1.2 Вычисление гомографии

После нахождения соответствий между особыми точками, производится вычисление примерного преобразования, связывающего два кадра видеопотока. Наиболее часто используемой моделью преобразования между двумя снимками является проективное преобразование плоскости или гомография. Преобразование имеет вид:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

где H — невырожденная матрица, определенная с точностью до масштаба. Для её определения нужны как минимум четыре соответствия между точками.

Для нахождения матрицы гомографии с учетом наличия ложных соответствий и шумов применяется метод Ransac [4], который представляет собой итеративный алгоритм оценки параметров модели на основе случайных выборок. На каждой итерации случайным образом выбирается минимальное количество точек, необходимое для построения модели, в данном случае это четыре точки. По выбранным точкам строится гипотеза H. Далее происходит проверка гипотезы: для всех соответствий вычисляется расстояние между реальным положением точки и гипотетическим. Для матрицы гомографии используется симметричное расстояние $\widetilde{d}(H, x, x') = \max(d(x', Hx), d(x, H^{-1}x'))$. Те точки, для которых $\widetilde{d}(H, x, x') < d_{\perp}$, считаются соответствующими гипотезе H. Итоговой признается гипотеза с наибольшим числом соответствующих ей точек.

Очень часто при построении панорамы ограничиваются нахождением гомографии между снимками и применением преобразования к одному из

Рис. 1: Панорама, полученная с использованием гомографии, $\Delta t = 80$ мс.

снимков. На рисунке 1 изображен результат применения такого подхода. Гомография идеально описывает движение камеры в двух случаях:

1. Камера вращается, оставляя неподвижным оптический центр. В этом случае $H = KRK^{-1}$, где K — матрица внутренних параметров камеры,а R — матрица поворота
2. Сцена представляет собой плоскость. Тогда $H = K(R - ta^T)$, где a — вектор нормали плоскости сцены, t — вектор перемещения камеры.

В остальных случаях возникает эффект параллакса, реальные координаты точки отклоняются от координат, прогнозируемых гомографией вдоль вектора направления движения обратно пропорционально расстоянию от соответствующей трехмерной точки до камеры $x' = KRK^{-1}x + \frac{Kt}{Z}$, что приводит к искажениям панорамы. Искажения можно уменьшить за счет увеличения частоты взятия кадров и, соответственно, уменьшения Δt, однако, как видно на рис. 1, этого не всегда бывает достаточно.

1.3 Content-preserving warping

Для компенсации искажений, возникающих из-за эффекта параллакса, применяется метод content-preserving warping. Впервые он использовал-

ся в работе [5] для решения задачи стабилизации видеопотока. Позже он стал использоваться также для построения панорам. В моей работе использовался немного измененный и расширенный для применения склейки множества изображений алгоритм, описанный в [6].

Метод предназначен для склейки двух изображений в одно, в моей работе он используется как составная часть алгоритма склейки кадров видеопоследовательности. Кадры последовательно добавляются в панораму путем склейки нового кадра с предыдущим кадром, для которого положение в панораме уже известно. Метод склейки двух кадров заключается в нахождении начального глобального преобразования, разбиении изображения путем введения равномерной сетки и наложении ячеек сетки на шаблонное изображение путем минимизации функционала энергии. Отличие используемого в данной работе алгоритма от алгоритма, предложенного в [6], заключается в способе построение глобальной гомографии, вместо способа, предложенного в статье, использовался метод Ransac.

Изображение, положение которого в панораме уже известно, будем называть шаблонным, а изображение, которое необходимо склеить с ним — входным.

Суть метода заключается в следующем: введем равномерную сетку I на входном изображении. Применив гомографию к узлам этой сетки, получим другую сетку $\overline{I}$. Однако сетка $\overline{I}$ не соответствует реальному положению сетки I в системе координат шаблонного изображения. Задача состоит в том, чтобы найти оптимальные координаты ячеек сетки $\hat{I}$ в системе шаблонного изображения, которые соответствуют координатам ячеек сетки I на входном изображении. Координаты оптимальной сетки производятся путем минимизации квадратичного функционала энергии, который состоит из следующих трех частей:

1. Слагаемое локального выравнивания. Отвечает за корректное наложение общей части двух изображений.

2. Слагаемое глобального выравнивания. Отвечает за наложение той части входного изображения, которая не видна на первом.

3. Слагаемое гладкости. Отвечает за то, чтобы ячейки сетки, имеющие большую цветовую дисперсию, лучше подходили друг к другу.

Кроме того, введем сетку $\widetilde{I}$, которая задает положение шаблонного изображения на панораме.

1.3.1 Локальное выравнивание

Ранее были найдены особые точки на двух изображениях и соответствия между ними. Локальное выравнивание основано на том, что особые точки переходят друг в друга. Для каждой такой точки на входном изображении находится ячейка сетки I в которой она располагается и вычисляются коэффициенты билинейной интерполяции особой точки относительно четырех координат узлов этой ячейки из соотношения

$$P_j = \sum_{i=1}^{4} a_{j,i} V_{j,i},$$

где P_j — координаты особой точки на входном изображении, $V_{j,i}$ — координаты узлов сетки, $a_{j,i}$ — коэффициенты билинейной интерполяции. Тогда, из предположения, что внутренние точки сеток I и $\widehat{I}$ связаны билинейным преобразованием, получаем, координаты особой точки входного изображения в системе шаблонного изображения можно записать как

$$\overline{P_j} = \sum_{i=1}^{4} a_{j,i} \widehat{V}_{j,i}$$

через координаты узлов сетки $\widehat{I}$. В качестве функционала локального выравнивания берется

$$E_p = \|\overline{P_j} - \widehat{P}_j\|^2 = \sum_{j=1}^{n} \| \sum a_{j,i} \widehat{V}_{j,i} - \widehat{P}_j\|^2. \quad (1)$$

Здесь $\widehat{P}_j$ — координаты особой точки шаблонного изображения в системе координат связанной с панорамой. Однако так как в рассматриваемой задаче требуется строить панораму для видеопотока, шаблонное изображение тоже имеет своё положение в панораме, задаваемое сеткой $\widetilde{I}$, и поэтому вместо координат особых точек в системе шаблонного изображения в формуле (1) следует использовать их координаты в сетке $\widetilde{I}$. Эти координаты получаются из соответствующих координат особой точки шаблонного изображения путем применения билинейного преобразования, определяемого

известными координатами сетки $\widetilde{I}$: $\widehat{P}_j = \sum_{i=1}^{4} b_{j,i}\widetilde{V}_{j,i}$, где $b_{j,i}$ — коэффициенты билинейной интерполяции ключевых точек шаблонного изображения в сетке $\widetilde{I}$.

Функционал можно записать в виде

$$E_p = \|A_p V - b_p\|^2,$$

где A_p имеет вид

$$\begin{pmatrix} \dots & 0 & a_{j,1} & 0 & 0 & \dots & a_{j,2} & 0 & 0 & \dots & a_{j,3} & 0 & 0 & \dots & a_{j,4} & 0 & 0 & \dots \\ \dots & 0 & 0 & a_{j,1} & 0 & \dots & 0 & a_{j,2} & 0 & \dots & 0 & a_{j,3} & 0 & \dots & 0 & a_{j,4} & 0 & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

$$b_p = \begin{pmatrix} \vdots \\ P_{j,x} \\ P_{j,y} \\ \vdots \end{pmatrix}.$$

1.3.2 Глобальное выравнивание

Для того, чтобы корректно наложить те части входного изображения, которые отсутствуют на шаблонном изображении, используется глобальное выравнивание. Предполагая, что гомография является хорошей аппроксимацией преобразования для этих областей, выбираем узлы ячеек $\widehat{I}$ как можно ближе к $\overline{I}$. Функционал имеет вид

$$E_g = \sum_i \tau_i \|\widehat{V}_i - \overline{V}_i\|^2,$$

где $\tau_i = 0$, если узел $\widehat{V}_i$ участвовал в локальном выравнивании и $\tau_i = 1$ в противном случае. В матричном виде функционал принимает вид

$$E_g = \|A_g V - b_g\|^2,$$

где

$$A_g = \begin{pmatrix} \dots & 0 & \tau_i & 0 & 0 & \dots \\ \dots & 0 & 0 & \tau_i & 0 & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

$$b_g = \begin{pmatrix} \vdots \\ \overline{V}_{i,x} \\ \overline{V}_{i,y} \\ \vdots \end{pmatrix}.$$

1.3.3 Слагаемое гладкости

Для дальнейшей минимизации локальной дисторсии и более точного наложения неоднородных областей в общий функционал добавляется слагаемое гладкости.

Рассмотрим треугольник $\overline{V}_1\overline{V}_2\overline{V}_3$. Его вершину $\overline{V}_1$ можно представить в локальной декартовой системе координат, образованной вершинами $\overline{V}_2$ и $\overline{V}_3$ как

$$\overline{V}_1 = \overline{V}_2 + u(\overline{V}_3 - \overline{V}_2) + vR(\overline{V}_3 - \overline{V}_2), R = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

где u и v — координаты $\overline{V}_1$ в данной системе, которые инвариантны к преобразованию подобия треугольника. Эти координаты могут быть легко вычислены из данного соотношения путем решения линейной системы. Для каждого треугольника слагаемое гладкости определяется как взвешенное расстояние между реальным положением вершины и предполагаемым при преобразовании подобия

$$E_s(\widehat{V}_i) = w_s\|\widehat{V}_1 - \widehat{V}_2 + u(\widehat{V}_3 - \widehat{V}_2) + vR(\widehat{V}_3 - \widehat{V}_2)\|^2,$$

где w_s — значение гладкости треугольника $\overline{V}_1\overline{V}_2\overline{V}_3$ — норма дисперсии значений пикселей в пространстве RGB внутри этого треугольника. В качестве функционала гладкости берется сумма функционалов для всех возможных треугольников. Каждый внутренний узел сетки участвует в восьми треугольниках, всего получается $8(nm - m - n + 1)$ треугольников, где m и n — количество узлов в сетке по строкам и по столбцам.

В матричном виде функционал имеет вид

$$E_s = \|A_s V\|^2,$$

где A_s имеет вид

$$\begin{pmatrix} \dots & 0 & -w_i & 0 & 0 & \dots & w_i(1-u) & -vw_i & 0 & \dots & uw_i & -vw_i & 0 & \dots \\ \dots & 0 & 0 & -w_i & 0 & \dots & vw_i & w_i(1-u) & 0 & \dots & vw_i & uw_i & 0 & \dots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

1.3.4 Результаты

Рис. 2: Панорама, полученная с помощью подхода content-preserving warping, $\Delta t = 800$.

Итоговый функционал является взвешенной суммой трех рассмотренных ранее функционалов.

$$E = E_p + \alpha E_g + \beta E_s.$$

В матричном виде

$$E = \|AV - b\|^2,$$

где

$$A = \begin{pmatrix} A_p \\ \alpha A_g \\ \beta A_s \end{pmatrix}, b = \begin{pmatrix} b_p \\ \alpha b_g \\ \mathbb{O} \end{pmatrix}.$$

Для нахождения оптимальных координат узлов сетки производится решение системы нормальных уравнений

$$A^T AV = A^T b.$$

После нахождения координат ячеек сетки производится построение панорамы, путем применения билинейных преобразований к внутренним точкам сетки. Найденная сетка $\widehat{I}$ сохраняется и используется на следующей итерации в качестве сетки шаблонного изображения $\widetilde{I}$. Эксперименты показали, что при $\alpha \in [0.02, 0.05]$ и $\beta = 0.001$ панорама получается наиболее качественной. Результат построения панорамы для фрагмента видеозаписи продолжительностью в 64 секунды изображен на рисунке 2. Сравнивая результат с результатом, полученным с помощью гомографии(рис. 1), видим, что полученная панорама лучше сохраняет реальную форму сцены.

Глава 2. Построение скелетных графов

Следующим этапом алгоритма является построение скелетных графов дорог для карты и склеенной ранее панорамы. Дороги, в особенности их части в окрестности перекрестков, используются в качестве признаков для наложения двух изображений друг на друга, поэтому особенно важно находить их как можно быстро и качественно.

2.1 Бинаризация изображений

2.1.1 Бинаризация карты

Бинаризация изображения карты производится посредством совмещения результатов глобальной и адаптивной бинаризации. Пиксель считается принадлежащим дороге, если оба алгоритма бинаризации отнесли пиксель к объекту. Такой подход позволяет компенсировать недостатки одного метода достоинствами другого.

Глобальная бинаризация производится посредством кластеризации значений яркостей пикселов, полагая число кластеров равным двум. В качестве алгоритма кластеризации использовался k-means [10]. Начальные центры кластеров выбирались случайным образом. Такой подход позволяет отделить более темные области, которые обычно соответствуют фону, от светлых, соответствующих объектам. Однако если два светлых объекта находятся близко друг к другу (например, дом белого цвета стоит рядом с дорогой), то алгоритм не сможет разделить эти два объекта и ошибочно примет их за один большой объект. Результат глобальной бинаризации изображения карты показан на рисунке 3.

Рис. 3: Результат глобальной бинаризации изображения карты.

Для адаптивной бинаризации использовался метод, описанный в [11]. Пороговое значение вычисляется для каждого пискела отдельно на основе информации об интенсивности соседних точек. С помощью интегрального изображения для каждого пиксела вычисляется m — среднее значение яркости в окне заданного размера. В качестве порога принимается значение $d = m(1 - \frac{t}{100})$, где t — максимальное отклонение от среднего в процентах. Этот метод хорошо работает в условиях неравномерной освещенности и позволяет отделить светлые объекты друг от друга. Однако он определяет много пикселов в темных однородных текстурах как принадлежащих объекту. Результат адаптивной бинаризации изображения карты показан на рисунке 4.

Рис. 4: Результат адаптивной бинаризации изображения карты.

Использование результатов двух алгоритмов одновременно позволяет в качестве объектов брать только светлые области и дает возможность отделить близкие по освещенности объекты друг от друга. После проведения бинаризации, находятся все компоненты связности бинаризованного изображения и те, которые содержат мало пикселов, удаляются. Цель этой операции — удаление шумов и мелких объектов. Окончательный результат бинаризации изображен на рисунке 5.

2.1.2 Бинаризация кадра видеопотока

Для бинаризации кадров видеопотока использовался гауссовский адаптивный метод бинаризации, реализованный в библиотеке OpenCV в функции adaptiveThreshold. Он отличается от метода, описанного в [11] тем, что вместо среднего значения яркости рассматривается взвешенная сумма значений яркостей m. В качестве весов выступают коэффициенты гаус-

Рис. 5: Результат совмещения глобальной и адаптивной бинаризации.

совского ядра с центром в рассматриваемом пикселе и параметром $\sigma = 0.3(0.5r - 1) + 0.8$, где r — радиус окна. В качестве порогового значения для пиксела выступает $m + c$, где c — заданная для всего изображения константа. Полученное бинарное изображение фильтруется с помощью медианного фильтра и морфологической операции открытия, после чего отбрасываются компоненты связности, содержащие меньше двух процентов пикселей от общего числа.

Полученные бинарные изображения склеиваются в панораму при помощи алгоритма, описанного в главе 1. В полученном панорамном изображении производится заполнение небольших по площади участков фона внутри объекта, поскольку они очень сильно влияют на топологический скелет области. Результат представлен на рисунке 6.

2.2 Нахождение топологического скелета области

Для построения топологического скелета использовался метод, описанный в [12]. Алгоритм обладает возможностью регулировать уровень чувствительности к шумам при помощи всего одного параметра, что делает его настройку довольно простой. Суть метода заключается в распространении волны от границ внутрь области и нахождении точек встречи нескольких волн.

Расстояние от точки области до границы находится путем решения краевой задачи

$$|\nabla T(x, y)| = 1,\ T|_\Gamma = 0. \tag{2}$$

Производится решение дискретного аналога уравнения краевой задачи (2) для нахождения значений функции $T_{i,j}$ для каждого пиксела. Дискретиза-

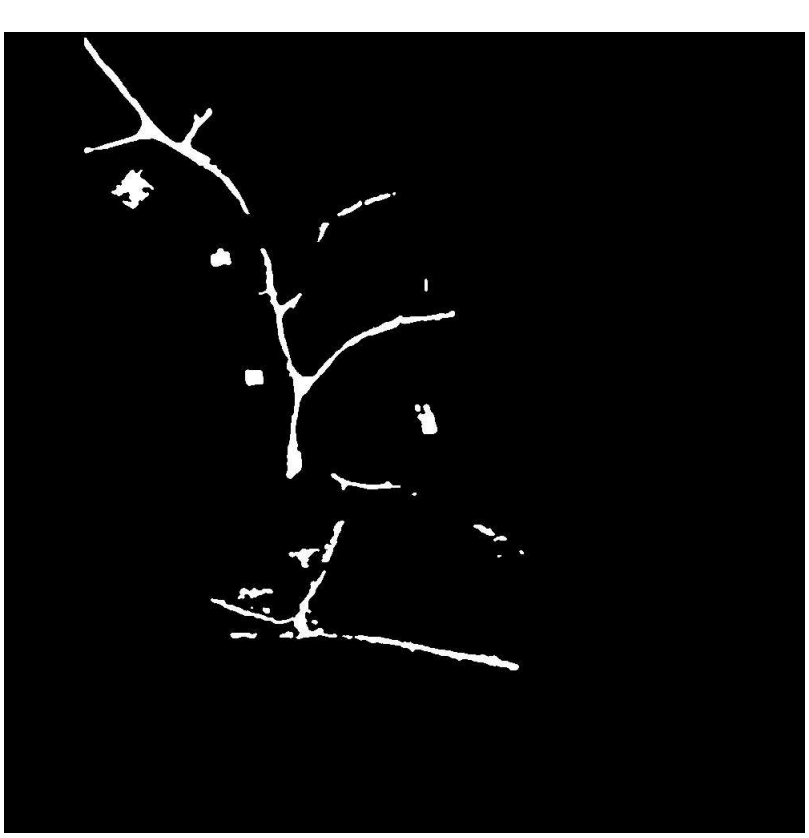


Рис. 6: Результат склейки бинаризованных кадров.

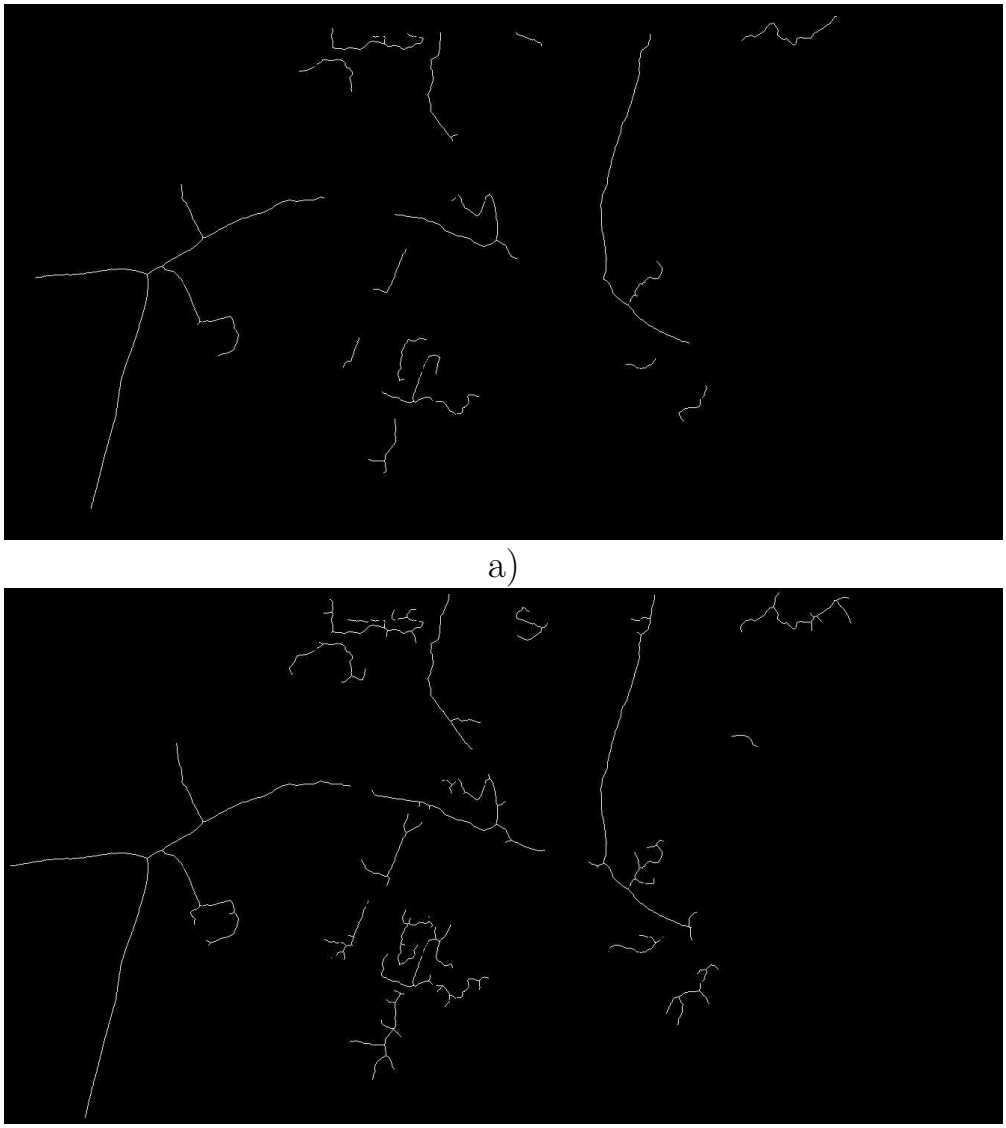
ция уравнения имеет вид

$$\max(D^{-x}T, -D^{x}T, 0)^2 + \max(D^{-y}T, -D^{y}T, 0)^2 = 1, \tag{3}$$

где $D^{-x}T_{i,j} = T_{i,j} - T_{i-1,j},\ D^{x}T_{i,j} = T_{i+1,j} - T_{i,j}$ и аналогично для направления y. Для нахождения значения $T_{i,j}$ применяется следующая процедура: для каждой соседней точки (k, l), происходит приближенное решение уравнения (3) для каждого из четырех квадрантов вокруг точки (k, l) и выбирается решение, дающее наименьшее значение T. Код функции, решающей уравнение для квадранта, приведен в приложении А.

Граничные точки нумеруются натуральными числами в порядке обхода контура начиная от произвольной точки. Вводится матрица $U_{i,j}$, которая для каждого пиксела содержит номер ближайшей к ней точки границы. Каждая точка помечается флагом $f_{i,j} \in \{Known, Band, Inside\}$, в зависимости от того, найдено ли для неё T, входит ли она в фронт волны или еще не рассмотрена. Для хранения точек фронта волны используется множество $NarrowBand$, которое представляет собой ассоциативный контейнер с возможностью быстрого извлечения минимального элемента. Изначально $NarrowBand$ содержит все точки границы области, для них же

$f_{i,j} = Band$. Для внутренних точек области $f_{i,j} = Inside$, а для внешних $f_{i,j} = Known$.



а)

б)

Рис. 7: Результаты нахождения топологического скелета: а) при $t = 100$; б) при $t = 15$.

Алгоритм состоит из следующих шагов:

1. До тех пор, пока $NarrowBand \neq \emptyset$, находим $T_{i,j}$— точку с наименьшим T из $NarrowBand$.

2. Для точек (k, l) из четырехсвязной окрестности (i, j) , у которых $f_{k,l} \neq Known$:

   (a) Если $f_{k,l} = Inside$, то помечаем $f_{k,l} = Band$ и вычисляем $U_{k,l}$

   (b) Обновляем значение $T_{k,l}$, путем решения уравнения (1) для 4-ех квадрантов вокруг (k, l), выбираем то решение, которое дает наименьшее T.

(c) Вставляем точку $T_{k,l}$ в $NarrowBand$.

В качестве точек скелета выбираются точки резкого разрыва U, то есть такие (i, j) что в её окрестности существует точка (k, l) такая что $|U_{i,j} - U_{k,l}| > t$, где t— пороговый параметр, отвечающий за чувствительность к шуму. Для удаления ложных веток скелета, которые возникают возле точек, в которых начинается отсчет точек границы, алгоритм запускается дважды для разных начальных точек и результирующий скелет является пересечением двух получившихся скелетов.

Как видно на рисунке 7, если взять порог t большим, то кроме удаления шума укорачиваются также и полезные ветви. Для устранения этого эффекта было предложено использовать два порога t_1 и $t_2, t_1 \gg t_2$. Точки скелета, полученного при $t = t_1$ сразу добавляются в результат, после чего начиная от концевых точек скелета происходит проход по точкам скелета с $t = t_2$ до тех пор пока не достигнут узел или конец. Таким образом, полезные ветви скелета не укорачиваются, а шум отсекается. Результат операции представлен на рисунке 8.

Рис. 8: Результат использования двух порогов.

2.3 Обработка скелетных графов

Рассмотренный выше алгоритм находит топологический скелет области в виде изображения. Для алгоритма наложения скелетов необходимо преобразовать скелет в форму графа, у которого ребра являются кривыми, заданными дискретно, а вершины — концевыми точками ребер. Перевод из

изображения в граф производится следующим образом:

1. На изображении находятся точки, у которых число соседних скелетных точек не равно двум, эти точки становятся вершинами графа.
2. Ребра добавляются проходом от одной вершины до другой по скелетному изображению, пройденные точки стираются из изображения.

После построения графа производится его фильтрация. Ко всем ребрам применяется фильтр Гаусса с радиусом окна равным 40. Для устранения граничных эффектов фильтрации ширина окна для точек вблизи концов сужается так, чтобы окно не выходило за границы кривой. После чего для всех компонент связности считается суммарная длина ребер и те компоненты связности, у которых она минимальна, удаляются. Так происходит удаление мелких объектов, таких как дома или деревья, и уменьшается количество вершин в графе. Полученные графы поступают на вход алгоритму нахождения преобразования между графами.

Глава 3. Наложение двух скелетных графов

После построения и соответствующей обработки скелетных графов необходимо найти примерное преобразование, связывающее два скелетных графа. В качестве модели преобразования было выбрано преобразование подобия, поскольку для случая, когда камера направлена строго вниз к земле, оно неплохо аппроксимирует возможные движения относительно карты. При нахождении оптимального преобразования возникают некоторые сложности. Во-первых, ребра на двух похожих скелетах могут претерпевать различные искажения, такие как разрывы одного ребра на два, локальные изменения кривизны ребер, обрезки части ребер. Во-вторых, на одном графе могут добавиться ребра и вершины, которые отсутствовали на другом графе. Поэтому возникает задача разработать алгоритм, который будет учитывать все эти особенности.

В ходе исследований был разработан алгоритм, ищущий оптимальное преобразование подобия в несколько этапов:

1. Вычисление признаков для всех вершин обоих скелетов.
2. Поиск попарных соответствий между всеми парами признаков и процедура их голосования для нахождения возможных ориентаций одного изображения относительно другого.
3. Для каждой ориентации вычисление сдвига и масштаба.

3.1 Вычисление признаков

В качестве признака вершины используется вектор, содержащий углы входа ребер в данную вершину, то есть углы между осью x и отрезком, аппроксимирующим ребро в окрестности вершины. В идеальном случае таким отрезком могла быть касательная к ребру в точке вершины, однако граф может быть искажен в окрестности вершины, поэтому возникает проблема нахождения этого отрезка. Для определения таких отрезков предложена следующая эвристика: последовательно происходит поиск подходящего отрезка длиной l, начиная от $l = l_{max}$ и уменьшение шага на Δl, если отрезок текущей длины не найден. Пусть Γ — ребро скелетного графа, a и b — индексы начала и конца отрезка, $a \geqslant 0$, $a < b$, изначально $a = 0,\ b = l_{max}$. Алгоритм состоит в следующем:

1. Находится $d_{max} = \max\limits_{i \in (0,l)} d(\Gamma_{i+a}, \overline{\Gamma_a \Gamma_b})$, где $d(\Gamma_k, \overline{\Gamma_a \Gamma_b})$ — расстояние от точки Γ_k до отрезка $\overline{\Gamma_a \Gamma_b}$.

2. Если $d_{max} \leqslant \varepsilon$, то $\overline{\Gamma_a \Gamma_b}$ — результирующий отрезок, иначе $a = a + \arg\max\limits_{i=0,\ldots,l} d(\Gamma_{i+a}, \overline{\Gamma_a \Gamma_b})$, $b = a + l$ и предыдущий шаг повторяется, пока не произойдет выход за пределы ребра. При достижении конца ребра уменьшаем l и начинаем проход сначала, пытаясь найти отрезок меньшей длины.

После нахождения координат концов отрезка, угол входа вычисляется в градусах по формуле

$$\theta = \frac{180}{\pi} \arctan\left(\frac{\Gamma_{b,y} - \Gamma_{a,y}}{\Gamma_{b,x} - \Gamma_{a,x}}\right) + 180, \theta \in [0, 360).$$

Для вычисления углов используется функция fastAtan2 библиотеки OpenCV, которая вычисляет полярный угол вектора. Все значения углов для каждой вершины сортируются по возрастанию. Также будем хранить координаты концов отрезков, они понадобятся на следующих этапах алгоритма.

3.2 Нахождение соответствий между вершинами

Пусть имеется две вершины скелета со степенями n и m, а θ_i^1, θ_j^2 — углы входа первой и второй вершины соответственно. Необходимо найти такие величины φ_i, что циклические последовательности θ^1 и $\theta^2 + \varphi_i$ в целом совпадут, то есть после удаления некоторого небольшого количества элементов из этих последовательностей и циклического сдвига одной из них, разница между соответствующими элементами этих последовательностей меньше некоторого наперед заданного значения. Введем функцию $\varrho(\psi_1, \psi_2)$ — разница между углами, $\varrho(\psi_1, \psi_2) = \psi_1 - \psi_2 + 360T$, где $T = \arg\min\limits_{T \in \mathbb{Z}} |\psi_1 - \psi_2 + 360T|$. Зафиксируем первую пару индексов соответствия (i, j). Вычислим $\psi = \theta_i^1 - \theta_j^2$. Далее будем последовательно искать для ребер первой вершины $t_1 = (i+1) \bmod n, \ldots, (i-1) \bmod n$ такое ребро второй вершины k, что $|\varrho(\psi, \theta_{t_1}^1 - \theta_k^2)| \leqslant 2\beta$, $k \in [(t_2 + 1) \bmod m, (j-1) \bmod m]$, где t_2 — индекс последнего добавленного в соответствие ребра второй вершины. Если для t_1 нашлось такое k, то пару (t_1, k) добавляем в соответствие и помечаем. Проделав вышеописанную процедуру для всех непомеченных

пар (i, j), получаем набор соответствий $h^l_{i,j}$, $l \in \{1, 2\}, i \in [1, s], j \in [1, r_i]$, где s — число найденных соответствий, а r_i — размер i-го соответствия, то есть число ребер в соответствии. Чтобы уменьшить число заведомо ложных соответствий, из набора соответствий оставляют только те, у которых $r_i \leqslant \frac{\max(n,m)}{2}$ и $i \in \arg\max\limits_{j \in [1,s]} r_j$. Углы соответствий φ_i вычисляются как решение задачи $\sum\limits_{j=1}^{r_i} (\varrho(\theta^1_{h^1_{i,j}}, \theta^2_{h^2_{i,j}}) - \varphi_i)^2 \to \min$. Её решением является

$$\varphi_i = \frac{\sum\limits_{j=1}^{r_i} \varrho\left(\theta^1_{h^1_{i,j}}, \theta^2_{h^2_{i,j}}\right)}{r_i}. \tag{4}$$

Код функции нахождения соответствий между двумя вершинами представлен в приложении Б.

3.3 Нахождение возможных ориентаций

Применив описанную выше процедуру для всех пар вершин первого и второго графов, получим набор соответствий H. Введем Λ — аккумулятор, $\Lambda_i \in \mathbb{N}$, $0 \leqslant i \leqslant 359$. Каждое соответствие H_i голосует за своё значение ориентации φ_i, оно добавляет в ячейки с $\Lambda_{\lfloor \varphi - \beta \rfloor}$ по $\Lambda_{\lceil \varphi + \beta \rceil}$ значение своего размера r_i. Вместе с тем запоминаем, какие соответствия куда вносят вклад.

Локальные максимумы в окне радиуса β берутся в качестве возможных ориентаций. Для каждого локального максимума происходит уточнение угла ориентации по формуле (4), примененной к проголосовавшим за эту ячейку соответствиям $M \subset H$. Далее, для каждой ориентации выполняется вычисление остальных параметров преобразования, в процессе которого некоторые соответствия отсеиваются. Итоговым преобразованием объявляется то, которому соответствует наибольшее число оставшихся соответствий.

3.4 Вычисление оставшихся параметров

Для вычисления оставшихся параметров используется метод RANSAC, описанный в [4] для вычисления оптимального преобразования, переводящего один набор параллельных прямых в другой. Пусть $l^k_{i,j}$ — прямые, проходящие через j-ый отрезок i-го соответствия из M на k-ом изобра-

жении. Введем $L^k = \{l^k_{0,0}, l^k_{0,1}, \ldots, l^k_{0,r_0}, l^k_{1,0}, \ldots, l^k_{s,0}, \ldots, l^k_{s,r_s}\}$, где $k = \{1, 2\}$ — наборы прямых. Чтобы добиться параллельности, прямые второго набора поворачиваются на угол ориентации для компенсации поворота, но даже после этого они пересекаются под углом меньшим β, чтобы это устранить, заменим направляющие косинусы прямых на их нормированную полусумму, то есть вместо прямых $L^1_i = \{a^1_i, b^1_i, c^1_i\}$ и $L^2_i = \{a^2_i, b^2_i, c^2_i\}$ возьмем прямые

$$L^1_i = \{\frac{a^1_i + a^2_i}{2}, \frac{b^1_i + b^2_i}{2}, c^1_i\},\ L^2_i = \{\frac{a^1_i + a^2_i}{2}, \frac{b^1_i + b^2_i}{2}, c^2_i\}, \tag{4}$$

здесь $\|(a^1_i, b^1_i)\| = 1,\ \|(a^2_i, b^2_i)\| = 1$.

Также добавим прямые, которые будут задавать связи между вершинами графов. Пусть после компенсации поворота координаты вершин имеют вид $V^k_i = \{V^k_{i,x}, V^k_{i,y}\}$, где $k = \{1, 2\},\ i \in [1, n_k]$. Координата V^k_i является точкой пересечения двух прямых, заданных уравнениями $x = V^k_{i,x}$ и $y = V^k_{i,y}$. Коэффициенты их уравнений имеют вид $l_x = \{1, 0, -V^k_{i,x}\},\ l_y = \{0, 1, -V^k_{i,y}\}$. Очевидно, что соответствующие прямые параллельны, поэтому мы можем добавить их в наборы. Такой подход позволяет получать полезную информацию от положения вершин даже при значительном искажении одной из координат вершины на втором изображении.

После вышеописанных процедур имеем два набора параллельных прямых с коэффициентами $L^1_i = \{a_i, b_i, c^1_i\}$ и $L^2_i = \{a_i, b_i, c^2_i\}, i = 1, \ldots, n$. После сдвига и масштабирования прямые первой группы переходят в прямые $L'_i = \{a_i s, b_i s, a_i t_x + b_i t_y + c^1_i\}$. Приравнивая коэффициенты L^1_i и L'_i с учетом однородности, получим уравнение $a_i t_x + b_i t_y - c^2_i s = -c^1_i$. Взяв три пары прямых, не пересекающихся в одной точке, получим систему

$$\begin{pmatrix} a_1 & b_1 & -c^2_1 \\ a_2 & b_2 & -c^2_2 \\ a_3 & b_3 & -c^2_3 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ s \end{pmatrix} = \begin{pmatrix} -c^1_1 \\ -c^1_2 \\ -c^1_3 \end{pmatrix},$$

которая имеет единственное решение, так как все три прямые не пересекаются в одной точке. Поскольку в постановке задачи гарантировалось наличие на местности хотя бы двух хорошо заметных перекрестков, то существует возможность выбрать три прямые так, чтобы они не пересекались в одной точке.

Производится некоторое количество итераций RANSAC с этой системой в качестве модели и пороговым параметром $d_\perp$. В качестве функции оценки соответствия данных гипотезе используется функция $d(l, l', T) = \max(d(Ta, l'), d(l, T^{-}1a'))$, где l, l' — рассматриваемая пара прямых,a, a'— точки, принадлежащие прямым l, l' до процедуры усреднения (4), T — гипотеза. После нахождения преобразования производится пересчет угла поворота по формуле (4) на прямых, удовлетворяющих гипотезе, и запуск RANSAC заново, до тех пор, пока гипотеза не стабилизируется.

Реальной объявляется ориентация, которой соответствует больше всего прямых, удовлетворяющих гипотезе.

Глава 4. Эксперименты

Для проведения эксперимента был взят фрагмент видеозаписи, продолжительностью в одну минуту и четыре секунды. В качестве карты были взяты несколько спутниковых изображений, взятых с сервисов Яндекс.Карты [13] и Google Карты [14]. Производилось наложение на карту панорам, произведенных двумя разными алгоритмами: при помощи гомографии при $\Delta t = 40$ мс и с помощью подхода content-preserving warping при $\Delta t = 800$ мс. Результаты представлены на рис. 10, 11. В ходе тестирования изображение карты поворачивалось на различные углы, результаты работы на некоторых повернутых изображениях представлены на рис. 12. На рис. 13 представлены результаты работы алгоритма для карты в виде схемы. Для всех представленных на рисунках тестов параметры алгоритма наложения двух снимков выбраны так: $\beta = 7$, $d_\perp = 16$, $l_{max} = 50$, $\varepsilon = 1$, $\Delta l = 5$. Таблицы 1 и 2 представляют результаты запуска программы с различными значениями параметров $d_\perp$ и β. На рис. 14 представлены несколько кадров видеопотока и результаты нахождения их позиций на карте.

Алгоритм наложения скелетных графов был дополнительно протестирован на некоторых изображениях из базы изображений Brown Univ Large Binary Image Database [15]. Были взяты одиннадцать изображений человечков и дополнительно создано еще два изображения на основе остальных, изображения представлены на рис. 9. Эксперимент заключался в следующем: все изображения поворачивались на случайный угол и применялось масштабирование со случайным коэффициентом из отрезка $[0.6, 1.5]$. После чего искались преобразования между всеми возможными парами скелетов картинок. Всего было проведено 100 таких экспериментов, их результаты представлены в таблице 3. Как видно, алгоритм верно находит преобразование в 88-99 процентах случаях, в зависимости от сложности изображений.

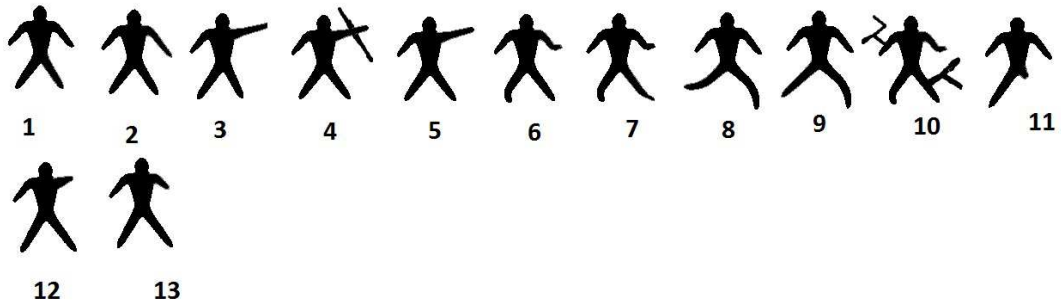


Рис. 9: Изображения из базы изображений Кимиа.

Рис. 10: Результат работы алгоритма наложения для панорамы, сделанной с помощью гомографии.

Рис.	$d_\perp$	4	8	12	16	20	24	28	32
12 в)	s	1.637	1.636	1.676	1.663	1.663	1.663	1.663	1.663
	θ	60.91	60.9	61.19	61.19	61.19	61.19	61.19	61.19
	Δx	890.6	890.7	894.2	892.9	892.9	892.9	892.9	892.9
	Δy	42.5	42.5	24.38	28.96	28.96	28.96	28.96	28.96
11 б)	s	1.394	1.676	1.698	1.684	1.687	1.389	1.31	1.302
	θ	301.5	299.5	299.5	301.2	301.5	301.5	301.5	301.5
	Δx	-70.91	-137.9	-144.4	-68.24	-51.95	-72.72	-54.26	-53.6
	Δy	495.91	508.3	512.4	482.48	471.6	467.6	447.2	449.3
13	s	1.70	3.51	3.32	3.33	3.51	3.33	3.50	3.39
	θ	120.8	300.7	301.4	300.8	300.8	300.8	300.8	300.3
	Δx	1044	-267.5	-230.6	-230.3	-272.2	-230.3	-272.2	-249.3
	Δy	290.6	938.7	905.3	914.9	938	914.9	940.1	920

Таблица 1: Запуск алгоритма при разных значениях $d_\perp$.

а)

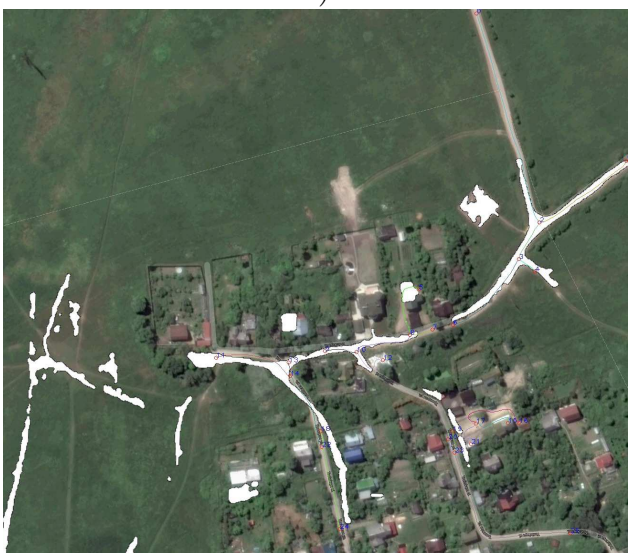
б)

Рис. 11: Результаты работы алгоритма наложения с панорамой, сделанной с использованием подхода content-preserving warping.

Рис.	β	4	5	6	7	8	9
12 в)	s	1.70	1.65	1.663	1.664	1.664	1.664
	θ	59.43	61.48	61.19	60.86	60.86	60.86
	Δx	880.54	895.23	892.96	889.84	889.84	889.84
	Δy	10.39	30.96	28.96	26.70	26.70	26.70
11 б)	s	1.453	1.50	1.69	1.7	1.68	1.39
	θ	299.73	300.49	298.8	301.44	299.61	299.68
	Δx	-86.50	-97.43	-97.59	-145.02	-114.36	-72.7
	Δy	492.73	493.81	487.92	514.03	499.48	480.67
13	s	0.69	2.84	3.32	3.47	3.6	3.6
	θ	75.03	351.9	301.3	301.6	303.1	303.1
	Δx	790.8	845.2	-229.9	-263.2	-299.3	-299.3
	Δy	208	278.6	905.1	920.3	916.5	916.5

Таблица 2: Запуск алгоритма при разных значениях β.

а)



б)

в)

Рис. 12: Результаты тестирования алгоритма при разных расположения карты.

Рис. 13: Результат работы алгоритма для карты схемы.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	—	0	0	3	2	0	4	2	0	2	0	1	4
2	0.019 0.85	—	0	1	1	4	2	2	0	4	2	0	0
3	0.018 0.81	0.016 0.85	—	3	3	2	3	7	6	10	2	0	2
4	0.018 0.92	0.016 0.98	0.024 0.98	—	0	6	8	9	5	12	11	1	6
5	0.013 0.95	0.016 1.1	0.022 1.1	0.012 0.65	—	0	4	3	6	3	5	0	2
6	0.037 1.2	0.03 1.3	0.027 1.5	0.035 0.67	0.037 0.82	—	4	3	2	3	1	2	1
7	0.028 1.2	0.029 1.2	0.025 1.4	0.033 0.86	0.034 0.79	0.021 0.97	—	4	4	3	0	2	1
8	0.016 1.4	0.029 1.3	0.022 1.3	0.027 1.2	0.023 1.1	0.032 1.1	0.024 1	—	0	6	2	5	9
9	0.018 1.2	0.02 1.2	0.024 1.4	0.026 1.2	0.027 1.1	0.02 1.3	0.026 1.2	0.013 0.75	—	9	1	4	5
10	0.039 1.1	0.031 1.4	0.045 1.4	0.034 0.74	0.034 0.79	0.016 0.81	0.02 0.85	0.031 1.1	0.025 1.1	—	3	2	0
11	0.023 1.1	0.043 1.3	0.033 1.4	0.035 0.87	0.038 0.93	0.026 0.72	0.025 0.97	0.016 1.1	0.022 1.1	0.036 0.81	—	0	0
12	0.057 1.2	0.059 1.4	0.04 1.4	0.071 0.87	0.07 1	0.038 0.72	0.024 1.1	0.021 1.4	0.022 1	0.037 0.96	0.029 1.1	—	0
13	0.058 1.7	0.065 2	0.038 1.9	0.071 1	0.07 0.93	0.043 1	0.033 1.1	0.036 1.3	0.035 1.1	0.037 1.1	0.026 0.96	0.018 1	—

Таблица 3: Результаты тестирования на изображениях из Brown Univ Large Binary Image Database. Значения выше главной диагонали — количество ошибочно найденных преобразований из 100. Значения ниже главной диагонали: верхнее — средний модуль отклонения по коэффициенту масштабирования найденных верно преобразований, нижнее — среднее отклонение по углу в градусах.

Рис. 14: Несколько кадров из видеопотока с найденными для них положениями на карте.

Анализ Результатов

Из рис. 10 и 11 панорама, построенная с помощью подхода content-preserving warping немного лучше соответствует изображению карты, чем построенная с помощью гомографии. Благодаря этому положение дальних от начала съемки кадров определяется с большей точностью.

Как видно из таблицы 3, алгоритм наложения скелетных графов может допускать ошибки в определении ориентации объекта. Ошибки могут возникать по нескольким причинам. Во-первых, в случае когда та часть графов, которая реально связана преобразованием подобия слишком мала по сравнению с остальной частью графа, которая претерпевает несвойственные для преобразования подобия изменения, например изменение угла, под которым ребра пересекаются в вершине. Во-вторых, в случае, когда на ребрах графа присутствует несколько разных участков, на которых ребро хорошо аппроксимируется прямой, алгоритм может выбрать в качестве признаков разные участки и вследствие чего дальнейшие шаги алгоритма не смогут сопоставить эти ребра друг с другом. Однако для графа дорог второй случай возникает редко, поскольку дороги очень часто близки по форме к прямым. Во избежание первой ситуации, перед процедурой наложения из графов удаляются мелкие ребра. В случае же верно найденной ориентации, алгоритм показывает неплохую точность.

Из таблиц 1 и 2 видно, что параметры $d_\perp$ и β должны быть достаточно большими для получения хорошего результата. При $\beta \leqslant 5$ алгоритм часто некорректно определяет ориентацию объекта, а при $d_\perp < 12$ масштаб и сдвиг часто определяется с большой ошибкой, поскольку в этом случае алгоритм хорошо накладывает отдельные части графа, а не граф целиком.

Как видно на рис. 14, положение далеких от места начала съемки кадров определяется с большей ошибкой, поскольку качество панорамы там немного хуже, чем для первых кадров. Для дальних кадров ячейки сетки, которая показывает положение кадра в панораме, сильнее отклоняются от квадрата, из-за чего найденные положения для этих кадров описываются уже произвольным многоугольником без самопересечений. Кроме того, свой вклад в общую ошибку также вносит ошибка наложения двух снимков. Однако даже в худшем случае для рассмотренной видеозаписи, общая ошибка не превышает четверти размера кадра.

Заключение

В ходе данной работы была разработана и реализована на языке C++ система сопоставления кадров видеопотока с цифровыми изображениями земной поверхности. В процессе её разработки были рассмотрены два способа построения панорамного изображения из множества кадров видеопотока, а также рассмотрен и реализован алгоритм нахождения топологического скелета бинарного изображения. Предложен алгоритм нахождения преобразования подобия между двумя скелетными графами, нечувствительный к разрывам и локальным искажениям ребер. Алгоритм рассматривает граф как набор локальных особенностей, поэтому его можно адаптировать к использованию также информации не только о дорогах, но и о зданиях и других объектах.

Список литературы

[1] Tian L., Kamata L., Ueshige Y., Kuroki Y. An automatic image-map registration algorithm using modified partial hausdorf distance // Proc. IEEE International Geoscience and Remote Sensing Symposium. 2005. Vol. 5. P. 3534–3537.

[2] Tian L., Kamata S. An automatic image-map alignment algorithm based on mutual information and Hilbert scan // Signal Processing Conference, 2008 16th European. IEEE, 2008. P. 1–5.

[3] Hild H. Automatic image-to-map-registration of remote sensing data, PhotogrammetrischeWoche // URL: http://elib. uni-stuttgart. de/opus/volltexte/2001/966. 2001.

[4] Fischler M. A., Bolles R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography // Communications of the ACM. 1981. Vol. 24, No 6. P. 381–395.

[5] Liu F., Gleicher M., Jin H., Agarwala A. Content-preserving warps for 3D video stabilization // ACM Transactions on Graphics (TOG). 2009. Vol. 28. No. 3. P. 44.

[6] Zhang F., Liu F. Parallax-tolerant image stitching // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. P. 3262–3269.

[7] Shi J., Tomasi C. Good features to track // Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE, 1994. P. 593-600.

[8] OpenCV. http://opencv.org (дата обращения: 20.04.17).

[9] Bouguet J. Y. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm // Intel Corporation. 2001. Vol. 5. No. 1–10. P. 4.

[10] Hartigan J. A., Wong M. A. Algorithm AS 136: A k-means clustering algorithm // Journal of the Royal Statistical Society. Series C (Applied Statistics). 1979. Vol. 28, No 1. P. 100–108.

[11] Bradley D., Roth G. Adaptive thresholding using the integral image // Journal of graphics, gpu, and game tools. 2007. Vol. 12, No 2. P. 13–21.

[12] Telea A., van Wijk J. J. An augmented Fast Marching Method for computing skeletons and centerlines // Data Visualization 2002. Proc. Eurographics – IEEE TCVG Symposium / Ed. by D. Ebert, P. Brunet, I. Navazo. New York: ACM, 2002. P. 251–259.

[13] Яндекс.Карты. https://yandex.ru/maps/ (дата обращения: 24.04.17).

[14] Google Карты. https://www.google.ru/maps (дата обращения: 24.04.17).

[15] Brown Univ Large Binary Image Database (Ben Kimia). http://vision.lems.brown.edu/sites/default/files/99db.tar.gz (дата обращения: 24.04.17).

Приложение А

Решение уравнения (1) для квадранта.

```
double SkeletonFastFMM::solve(int i1,int j1,int i2,int j2)
    {
        if(i1<0||i1>=T.rows||j1<0||j1>=T.cols)
            return MAX_VAL;
        if(i2<0||i2>=T.rows||j2<0||j2>=T.cols)
            return MAX_VAL;
        double r,s;
        double t1=T.at(i1,j1);
        double t2=T.at(i2,j2);
        double maxt=std::max(t1,t2);
        if(state.at(i1,j1)==CellState::KNOWN)
            if(state.at(i2,j2)==CellState::KNOWN)
            {
             r=sqrt(2-(t1-t2)*(t1-t2));
                s=(t1+t2-r)/2;
                if(s>maxt) return s;
                else
                {
                    s+=r;
                    if(s>maxt) return s;
                }
            }
            else return 1+t1;
        else if(state.at(i2,j2)==CellState::KNOWN)
            return 1+t2;
        return MAX_VAL;
    }
```

Приложение Б

Нахождение соответствий между двумя вершинами скелетного графа.

```
void SkeletonMatcher::distance(const Feature& feature1,
    const Feature& feature2,std::vector<MatchInfo>& res,
    const int angtol) const
{
static cv::Mat used(10,10,CV_8U);
int maxsize=0;
used.setTo(0);
const int start=res.size();
const int minn=
int(std::max<double>(feature1.size(),feature2.size())/2.0+0.5);
for(int i=0;i<feature1.size();i++)
    for(int j=0;j<feature2.size();j++)
        if(!used.at<uchar>(i,j))
        {
        int t1=i,t2=(j+1)%feature2.size();
        double ang=feature1[i]-feature2[j];
        MatchInfo info(&feature1,&feature2,0);
        info.matches.push_back(std::make_pair(i,j));
        used.at<uchar>(i,j)=1;
        double diff=ang;
        double dist=0;
            do
            {
            t1=(t1+1)%feature1.size();
            for(int k=t2;k!=j;k=(k+1)%feature2.size())
            {
            double d=feature1[t1]-feature2[k];
            while(abs(d-ang)>abs(d-ang+360))
                d+=360;
            while(abs(d-ang)>abs(d-ang-360))
                d-=360;
```

```
            if(abs(d-ang)<angtol)
                {
                diff+=d;
                used.at<uchar>(t1,k)=1;
                info.matches.push_back(std::make_pair(t1,k));
                t2=(k+1)%feature2.size();
                break;
                }
            }
info.angle=diff/info.matches.size();
        }
while((t1!=i)&&(t2!=j));
        if(info.matches.size()>=minn)
        {
         res.push_back(info);
            maxsize=std::max<int>(maxsize,info.matches.size());
         }
    }
    res.erase(
    std::remove_if(res.begin()+start,res.end(),
    [maxsize](MatchInfo& info)->bool{
     return maxsize>info.matches.size();
     }),res.end());
}
```